**Tapio Koskinen 1800509**

**Exercise 2** – **Tapio Koskinen** (Remember to fill your first and last name both here and in the file name)

1. Explain the following terms:
    a. Pseudocode:
    The definition of pseudo is “not genuine”. The idea behind pseudocode is that it can illustrate the ideas for a function, program, structure etc. Pseudocode contains the logic, but not the syntax of a real program.
    b. Algorithm:
    Algorithm is a set of instructions designed to achieve a solution to a problem. It can be simple or complicated. An example would be “grab a book from shelf”. Do you know where it is? -> take it, if not-> figure out where it is.
    c. Data attribute
    Data attribute represents characteristics or features of a data object. Attribute types can vary, but they all tell something about the object, for example and object can have a data attribute color that could have “yellow” as a value, or gender with a value “alien”
    d. Method
    Method is a set of instructions that can perform a task and it is called via an object of a class. Unlike a function, it requires a class to work.

2. Pseudocode

```
Exercise 2 - Part 2

Pseudocode

NumberOfExercises = user input

if NumberOfExercises is less than 9
    print out "Fail"

else if NumberOfExercises is 9
    print out "Your grade is 1"

else if NumberOfExercises is 10
    print out "Your grade is 2"

else if NumberOfExercises is 11
    print out "Your grade is 3"

else if NumberOfExercises is 12
    print out "Your grade is 4"

else if NumberOfExercises is 13
    print out "Your grade is 5"
```

**Tapio Koskinen 1800509**

3. Code

```
#Exercise 2 - Part 3
#Tapio Koskinen

asking = True
while asking:
    try:
        number_of_exercises = int(input("How many exercises did you do: "))
        break
    except ValueError:
        continue


grade_book = {9:1,10:2,11:3,12:4,13:5}

if number_of_exercises < 9:
    print("You did not pass the course")
elif number_of_exercises > 13:
    print("Your grade is 5")
else:
    print("Your grade is:",grade_book[number_of_exercises])
```

output

```
= RESTART: C:\[redacted]\Kevä
\CodeFiles\part3.py
How many exercises did you do: 11
Your grade is: 3
>>>
= RESTART: C:\[redacted]\Kevä
\CodeFiles\part3.py
How many exercises did you do: 4
You did not pass the course
>>>
= RESTART: C:\[redacted]\Kevä
\CodeFiles\part3.py
How many exercises did you do: -2
You did not pass the course
>>>
= RESTART: C:\[redacted]\Kevä
\CodeFiles\part3.py
How many exercises did you do: 25
Your grade is 5
>>>
```

**Tapio Koskinen 1800509**

4. Pseudo

```
Part 4 pseudocode


student list = a list
student grade list = a list

Student list = User inputs name
Sudent grade list = User inputs grade

the_average = sum of grades in grade list/lenght of student list

print out: "the average score of the class is [the_average]"
```

**Tapio Koskinen 1800509**

5. Code

```
#Part 5 - The average score
#Tapio Koskinen


#Create a function that creates a dictionary of students and
#their grades
def student_grade_dictionary():
    student_list = {}

    asking = True
    while asking:
        #Student name will be the key
        student = input("Student name:")
        #Student grade will be the value of current key
        student_list[student] = int(input("Grade:"))
        add_more = input("Add more?(y/n): ")
        if add_more == "y":
            continue
        elif add_more == "n":
            break
    #Return the created class
    return student_list

def average_grade(your_dict):
    #A placeholder for the sum of grades
    sum_of_grades = 0
    #Loop through all the students in the dictionary and
    #sum their grades
    for key in your_dict:
        sum_of_grades += your_dict[key]
    #The average will be the sum divided to the sum of students
    average = sum_of_grades/len(your_dict)
    #Return that value
    return average

def main():
    #Create your classroom
    print("Create your class with grades\n")
    my_students = student_grade_dictionary()
    #Print out the average grade of your classroom
    print("The average grade of this class is",average_grade(my_students))
#Run the program
main()
```

Output

```
= RESTART: [redacted]\Kevät2021\Obj
\CodeFiles\part5.py
Create your class with grades

Student name:Maija
Grade:4
Add more?(y/n): y
Student name:Mikko
Grade:2
Add more?(y/n): y
Student name:Liisa
Grade:5
Add more?(y/n): y
Student name:Lauri
Grade:3
Add more?(y/n): n
The average grade of this class is 3.5
>>>
```

6. I'm not sure if I'm right, but I would consider having 5 attributes: actual time(this could be divided into hours minutes and seconds, but that would be unnecessary work), time of alarm(same as previous), state of the alarm(boolean), ringing(boolean) and a displayscreen. Everything is restricted except setting the alarm. Methods included would be users input for the time and weather it is on or off and these would also be public.

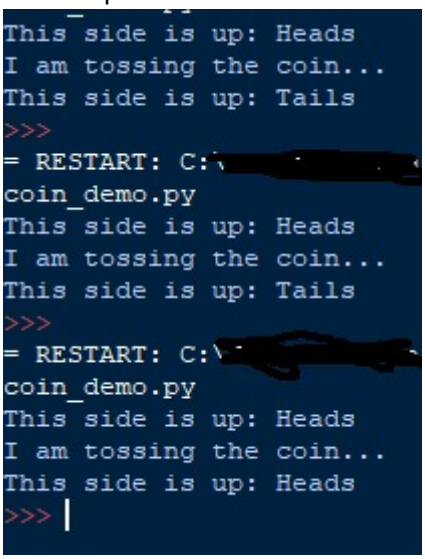
**Tapio Koskinen 1800509**

7. Coin code

```
"""
Created on mon jan 18 12:40 2021
@File:
@Description: The Coin simulates a coin that can be flipped
@author: tkoskinen

"""

import random

class Coin:

    # The __init__ method initializes the sideup data attribute with heads

    def __init__(self):
        self.sideup = 'Heads'

    # The toss method generate a random number
    # in the range of 0 through 1. If the number is
    # 0, then sideup is set to 'Heads', otherwise sideup
    # is set to 'Tails'.

    def toss(self):

        if random.randint(0,1) == 0:
            self.sideup = 'Heads'
        else:
            self.sideup = 'Tails'

    # The get_sideup method returns the value referenced by sideup

    def get_sideup(self):
        return self.sideup


# The main function

def main():

    # Create an object from the Coin class.
    my_coin = Coin()

    # Display the side of the coin that is facing up.
    print('This side is up:', my_coin.get_sideup())

    #Toss the coin.
    print('I am tossing the coin...')
    my_coin.toss()

    # Display the side of the coin that is facing up.
    print('This side is up:', my_coin.get_sideup())

# Call the main function
main()
```

**Tapio Koskinen 1800509**

Coin output



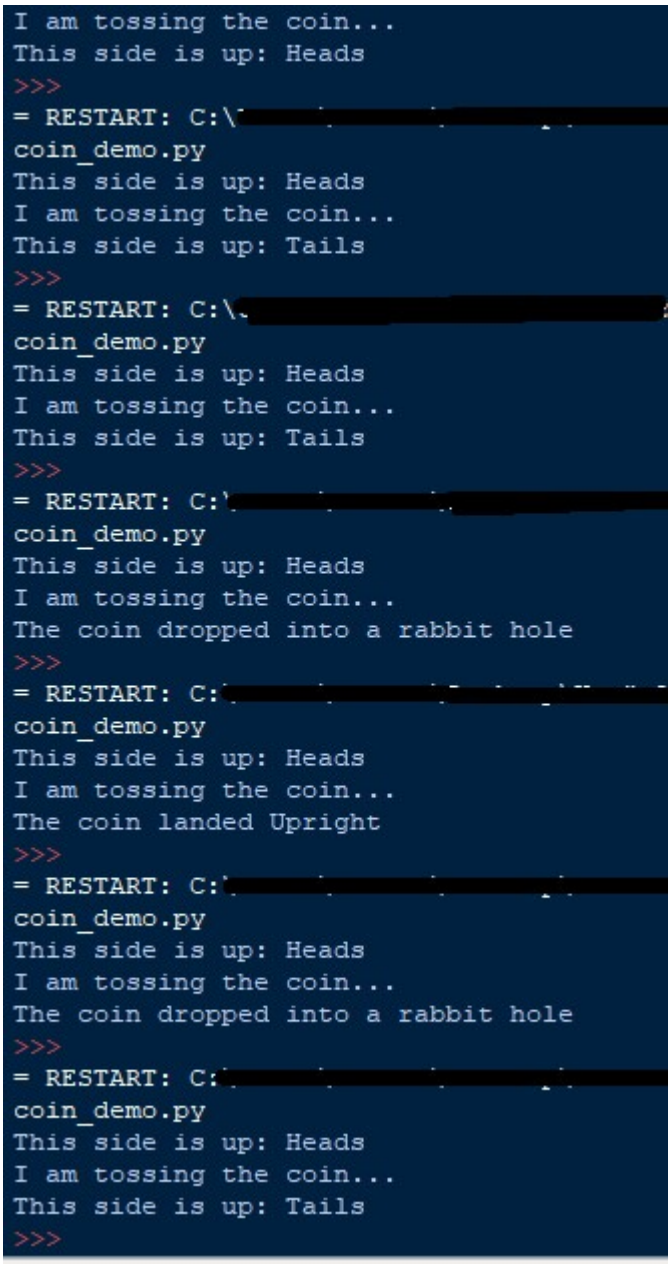
**Tapio Koskinen 1800509**

8. Modified coin code

```
    def toss(self):


        index = random.randint(0,3)

        #Not so realistically, all have equal chances of happening.

        if index == 0:
            self.sideup = 'Heads'
        elif index == 1:
            self.sideup = 'Tails'
            #Coin lands upright
        elif index == 2:
            self.sideup = 'Upright'
            #Coin flies off
        elif index == 3:
            self.sideup = 'None'

    # The get_sideup method returns the value referenced by sideup

    def get_sideup(self):
        return self.sideup


# The main function

def main():

    # Create an object from the Coin class.
    my_coin = Coin()

    # Display the side of the coin that is facing up.
    print('This side is up:', my_coin.get_sideup())

    #Toss the coin.
    print('I am tossing the coin...')
    my_coin.toss()

    # Display the side of the coin that is facing up.
    if my_coin.get_sideup() == 'Heads' or my_coin.get_sideup() == 'Tails':
        print('This side is up:', my_coin.get_sideup())

    #if the value is Upright print another message
    elif my_coin.get_sideup() == 'Upright':
        print('The coin landed', my_coin.get_sideup())

    #if the value is None, print the following text
    elif my_coin.get_sideup() == 'None':
        print("The coin dropped into a rabbit hole")

# Call the main function
main()
```

Modified coin output

**Tapio Koskinen 1800509**



**Tapio Koskinen 1800509**

9. Pseudocode for an alarm clock

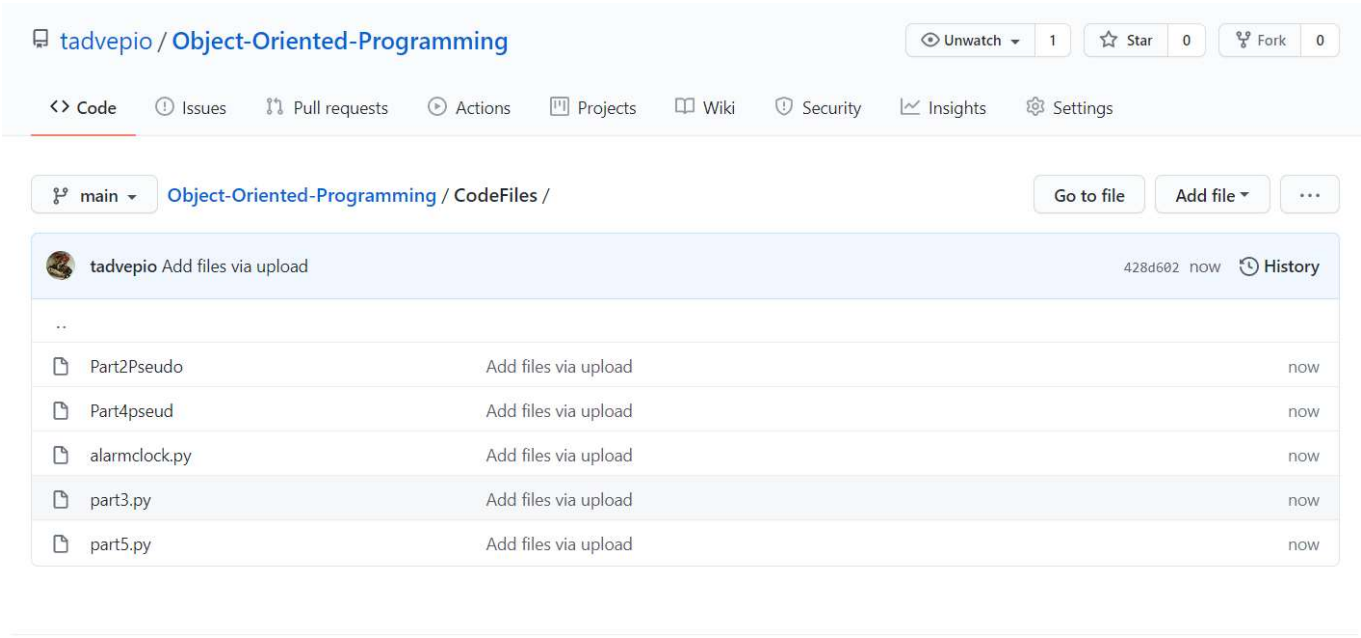
```
Pseudocode for alarmclock

------------------------

import necessary modules


define class clock:

    define actual time:

        real time = time in hour minute second
        return real time


    define alarm state:

        on or off = user input boolean
        return on or off


    define alarm(alarm time, alarm state) method:

        if alarm state is true

            while looping

                current time = time in hours, minutes and seconds

                if alarm time == current time:

                    print out: "Wake up!"

                    ringing sound

                    break out of loop


    define alarm time method:

        time = user input(hour,minute,second)

        return time
```

**Tapio Koskinen 1800509**

```
define the main program:

    tkinter widget

    loop with while

        time to gui:

            clock actual time

        display alarm setting:

            clock alarm time
            clock alarm state

        alarm method()


main program
```

10. Alarm clock
    I can't get the code to work similarly as in the pseudocode.

Screen capture of the Git status:



**Tapio Koskinen 1800509**

Self-assessment:

The alam clock task was by far hardest for me. I'm still unsure what object oriented programming is and making classes and objects limits the way I can code. I felt like the alarm clock task was way harder than the 8 other tasks. I'm not sure how to implement tkinter as an object or a class so I gave up after hours of problemsolving. I atleast got some training on tkinter. I'm disappointed on myself on this one.